Statistical Techniques in Robotics (16-831, S20) Lecture #14 (Wednesday, March 24)

EXP3 and EXP4

Lecturer: Kris Kitani *Scribes: Akshay Dharmavaram, Alex Withers*

1 Review

1.1 Thompson Sampling

In the last lecture, we learned about the Thompson sampling and the Beta-Bernoulli Bandit algorithms. We will first review the main concept of the multi-arm bandit problem, which is to maximize our payoff by selecting the appropriate arm, conditioned on the observed history. The Thompson Sampling algorithm is a multi-arm bandit problem that falls under the category of Context-free algorithms for stochastic environments. It is also known as the probability matching strategy. In this approach, we aim to balance exploration and exploitation by conditioning our policy to maximize the reward with respect to a set of belief variables.

In the stochastic environment framework, the rewards are not deterministic, and are approximated as a probability distribution. The Thompson Sampling approach parameterizes this distribution by defining θ, a prior over theta $p(\theta)$, and a reward distribution $r \sim p(r \mid a, \theta)$. Our goal is to maintain a running estimate of the prior based on the observed history. We select the arm based on the following:

$$a = \underset{k}{\operatorname{argmax}} \mathbb{E}_{p(r|a_k,\theta*_k)} \left[r \mid a_k, \theta*_k \right] \tag{1}$$

As the prior is something that needs to be learned, we compute it based using the observed history and take the argmax to obtain the posterior distribution of θ as:

$$p\left(\theta \mid h^{(t)}\right) = p\left(\theta \mid a^{(1)}, r^{(1)}, \ldots, a^{(t)}, r^{(t)}\right) \tag{2}$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\ p\left(\theta \mid h^{(t)}\right) \tag{3}$$

As it is non-trivial to model the distribution without any further constraints, we impose the Markov assumption and use the Bayes' Rule to simplify the equations, as follows.

$$p\left(\theta \mid a^{(1)}, r^{(1)}, \ldots, a^{(t)}, r^{(t)}\right) = \frac{p\left(a^{(1)}, r^{(1)}, \ldots, a^{(t)}, r^{(t)} \mid \theta\right) p(\theta)}{p\left(a^{(1)}, r^{(1)}, \ldots, a^{(t)}, r^{(t)}\right)} \tag{4}$$

$$p\left(a^{(1)}, r^{(1)}, \ldots, a^{(t)}, r^{(t)} \mid \theta\right) \to p\left(r^{(t)} \mid a^{(t)}, \theta\right) p\left(a^{(t)}\right) \ldots p\left(r^{(t)} \mid a^{(t)}, \theta\right) p\left(a^{(1)}\right) \tag{5}$$

We substitute these equations into the equations 2 and 3, to obtain the following:

$$p\left(\theta \mid h^{(t)}\right) = \frac{p\left(h^{(t)}p(\theta)\right.}{p\left(h^{(t)}\right)} \propto \prod_t p\left(r^{(t)} \mid a^{(t)}, \theta\right) p(\theta) \tag{6}$$

$$\hat{\theta} = \underset{\theta_k}{\operatorname{argmax}}\ p\left(r^{(t)} \mid a_k^{(t)}, \theta_k\right) p\left(\theta_k \mid h_k^{(t-1)}\right) \tag{7}$$

1.2 Parameterization using Beta and Bernoulli distributions

We parametrize our prior and posterior distributions to obtain efficient updates, or updates that would require minimal further computation. Thus, we choose conjugate distributions as our prior and posterior approximations due to their special property of conjugation. Conjugate distributions or conjugate pairs refer to a pair of posterior distributions and prior distributions for which the resulting posterior distribution belongs into the same parametric family of distributions of the prior distribution. Due to reflexivity, the prior distribution is also a conjugate prior for this sampling distribution.

It is known that Beta and Bernoulli distributions are conjugate pairs, as defined below:

The Bernoulli distribution is defined as

$$p(r \mid \theta) = \theta^r (1-\theta)^{1-r} \tag{8}$$

The Beta distribution is defined as:

$$p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{(\alpha-1)} (1-\theta)^{\beta-1} \tag{9}$$

$$\Gamma(n) = (n-1)! \tag{10}$$

We can reduce the posterior as follows:

$$p(r \mid \theta) = \theta^r (1-\theta)^{1-r} \tag{11}$$

$$p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{(\alpha-1)} (1-\theta)^{\beta-1} \tag{12}$$

$$\tag{13}$$

$$p\left(\theta \mid h^{(t)}\right) \propto p(r \mid a, \theta) p\left(\theta \mid h^{(t-1)}\right) \tag{14}$$

$$\propto \theta^r (1-\theta)^{(1-r)} \theta^{\alpha-1} (1-\theta)^{(\beta-1)} \tag{15}$$

$$\propto \theta^{r+\alpha-1} (1-\theta)^{1-r+\beta-1} \tag{16}$$

$$\propto \theta^{(r+\alpha)-1} (1-\theta)^{(1-r+\beta)-1} \tag{17}$$

$$\propto \theta^{\alpha'-1} (1-\theta)^{\beta'-1} \tag{18}$$

Finally, the posterior updates can be defined as follows:

$$\beta' = \beta + 1 - r \tag{19}$$
$$\alpha' = \alpha + r \tag{20}$$

1.3 Thompson Sampling & Bernoulli-Beta Thompson Sampling

Putting everything to gether, we can define the Thompson Sampling & Bernoulli-Beta Thompson Sampling algorithms. We also present their regret bound, $BR(T) = O(\sqrt{KT \log T})$.

Algorithm 1 Thompson Sampling Pseudo-Code

```
1: for $t = 1, \cdots, T$ do
2:     $\theta_k \sim p\left(\theta_k \mid h_k^{(t')}\right) \forall k$
3:     $a_{\hat{k}}^{(t)} = \text{argmax}_k \, \mathbb{E}_{p(r|a_k,\theta_k)}\left[r \mid a_k, \theta_k\right]$
4:     RECEIVE $(r^{(t)} \in \{0,1\})$
5:     $h_{\hat{k}}^{(t'+1)} = h_{\hat{k}}^{(t')}\left(r^{(t)}, a_{\hat{k}}^{(t)}\right)$
6:     $p\left(\theta_{\hat{k}} \mid h_{\hat{k}}^{(t'+1)}\right) \propto p\left(h_{\hat{k}}^{(t')} \mid \theta_{\hat{k}}\right) p\left(\theta_{\hat{k}} \mid h_{\hat{k}}^{(t')}\right)$
7: end for
```

Algorithm 2 Bernoulli-Beta Thompson Sampling Pseudo-Code

```
1: for $t = 1, \cdots, T$ do
2:     $\theta_k \sim p\left(\theta \mid \alpha_k, \beta_k\right) \forall k$
3:     $a_{\hat{k}}^{(t)} = \text{argmax}_k \, \mathbb{E}_{p(r|a_k,\theta_k)}\left[r \mid a_k, \theta_k\right]$
4:     RECEIVE $(r^{(t)} \in \{0,1\})$
5:     $\alpha_{\hat{k}} = \alpha_{\hat{k}} + r^{(t)}$
6:     $\beta_{\hat{k}} = \beta_{\hat{k}} + 1 - r^{(t)}$
7: end for
```

2 Summary

2.1 Adversarial Environments

In the previous sections, we have discussed two context-free, multi-arm bandit algorithms for stochastic environments. Now, we will move on to the adversarial environments, and discuss one context-free algorithm and one contextual bandit algorithm.

The adversarial bandit environment is defined as an arbitrary series of reward vectors that precisely define the outcome of the arms at any given time-step. An intuitive way to comprehend this is by visualizing a matrix, where the rows correspond to the arm rewards and the columns correspond to the reward values for a given time-step. Thus, to obtain the reward for an arm at a given time-step, we would need to retrieve the reward value in the matrix corresponding to the appropriate arm and time-step. The reason such a type of environment is termed as adversarial, is because the worst case scenario for the arbitrarily defined sequence of reward vectors, could be the adversarial case for a given algorithm.

The adversarial case is comparatively arbitrary, when compared to the stochastic environment. In the stochastic environment, we assume that there is a certain underlying phenomenon describing the rewards. However, in the adversarial case, we assume that the worst case scenario is that the environment is adversarial with respect to our chosen algorithm. This is not uncommon, as many phenomena in nature are too complex to model, thereby seeming arbitrary. Thus, we would need to engineer reactionary algorithms that learn to pull the best possible arm, when considering the history in hindsight.

Thus, the goal for an ideal multi-arm bandit algorithm, for an adversarial environment, will be to minimize the regret, regardless of the reward sequence.

2.2 Hedge Algorithm

The hedge algorithm is a PWEA algorithm that approximates a weight distribution that conditionally defines a policy. The pseudo-code is written as follows.

Algorithm 3 Hedge Algorithm

```
1: $\mathbf{w}^{(1)} \leftarrow \{w_k^{(1)} = 1\}_{k=1}^{K}$                         ▷ Weight initialization
2: for $t = 1, \cdots, T$ do
3:     Receive $(x^{(t)} \in \{-1, 1\})$                                      ▷ Receive reward
4:     $\mathbf{p}^{(t)} = \frac{\mathbf{w}^{(t)}}{\sum_k w_k^{(t)}}$                    ▷ probability over actions
5:     $k \sim$Multinomial $(\mathbf{p}^{(t)})$                                   ▷ randomly choose action
6:     $\hat{y}^{(t)} = h_k(x^{(t)})$                                           ▷ take action
7:     Receive $(r^{(t)} \in \{-1, 1\})$                                      ▷ Receive reward
8:     $w_k^{(t+1)} = w_k^{(t)} \exp\left(-\beta\left(\mathbf{1}\left[y^{(t)} \neq h_n\left(x^{(t)}\right)\right]\right)\right)$   ▷ Weight update
9: end for
```

2.3 Estimators

2.3.1 Biased Estimator

An estimator $\hat{X}$ of a variable X is unbiased if:

$$E\left[\hat{X}\right] = X \tag{21}$$

We will define a hypothetical custom reward estimator as follows:

$$c_i^{(t)}\left(a^{(t)}\right) = \mathbf{1}\left[a^{(t)} = i\right] \cdot r^{(t)} \tag{22}$$

Using the definition for an unbiased estimator, we plug in our estimator, for the single arm case, into the definition.

$$\mathbb{E}_{p(a)}\left[c_1^{(t)}\left(a^{(t)}\right)\right] = p\left(a^{(t)}=1\right)\cdot \mathbf{1}\left[a^{(t)}=1\right]\cdot r^{(t)} \tag{23}$$
$$= r^{(t)} \tag{24}$$

We also notice that over T time-steps, the expectation of our estimator is:

$$\mathbb{E}_{p(a)}\left[\frac{1}{T}\sum_{t=1}^{T} c_1^{(t)}\left(a^{(t)}\right)\right] = \frac{1}{T}\sum_{t=1}^{T} r^{(t)} = r^{(t)} \tag{25}$$

We notice that for a single arm, our estimator is indeed unbiased.

Now, we will show that for the multi-arm case, that the estimator is biased. We will show the counter example for the two-arm case here. We follow the same procedure as shown above.

$$\mathbb{E}_{p(a)}\left[c_1^{(t)}\left(a^{(t)}\right)\right] = p\left(a^{(t)}=1\right)\cdot c_1^{(t)}(1) + p\left(a^{(t)}=2\right)\cdot c_1^{(t)}(2) \tag{26}$$
$$= p\left(a^{(t)}=1\right)\mathbf{1}\left[a^{(t)}=1\right]\cdot r^{(t)} + p\left(a^{(t)}=2\right)\mathbf{1}\left[a^{(t)}=1\right]\cdot r^{(t)} \tag{27}$$
$$= (0.5)\cdot 1\cdot r^{(t)} + (0.5)\cdot 0\cdot r^{(t)} \tag{28}$$
$$= (0.5)\cdot r^{(t)} \tag{29}$$

We can also see that for the accumulating reward condition, we get:

$$\mathbb{E}_{p(a)}\left[\frac{1}{T}\sum_{t=1}^{T} c_1^{(t)}\left(a^{(t)}\right)\right] = \frac{1}{T}\sum_{t=1}^{T}(0.5)r^{(t)} \tag{30}$$

Thus, we show that the estimator is biased.

2.3.2 Unbiased Estimator

For our frameworks, we will define a custom unbiased reward estimator as follows:

$$c_i^{(t)}\left(a^{(t)}\right) = \mathbf{1}\left[a^{(t)}=i\right]\cdot r^{(t)}\cdot \frac{1}{p_i} \tag{31}$$

Now, we will show that for the multi-arm case, that the estimator is unbiased. We will show the proof for the two-arm case here, which can be easily extended to the multi-arm case. We follow the same procedure as shown above.

$$\mathbb{E}_{p(a)}\left[c_1^{(t)}\left(a^{(t)}\right)\right] = p\left(a^{(t)}=1\right)\cdot c_1^{(t)}(1) + p\left(a^{(t)}=2\right)\cdot c_1^{(t)}(2) \tag{32}$$
$$= p\left(a^{(t)}=1\right)\mathbf{1}\left[a^{(t)}=1\right]\cdot r^{(t)}\cdot\frac{1}{p_1} + p\left(a^{(t)}=2\right)\mathbf{1}\left[a^{(t)}=1\right]\cdot r^{(t)}\cdot\frac{1}{p_1} \tag{33}$$
$$= (0.5)\cdot 1\cdot r^{(t)}\cdot\frac{1}{0.5} + (0.5)\cdot 0\cdot r^{(t)}\cdot\frac{1}{0.5} \tag{34}$$
$$= r^{(t)} \tag{35}$$

Thus, we show that the estimator is unbiased.

2.4 EXP3

The EXP3 algorithm is listed as Algorithm 4 below. The update equation on line 7 states that a weight is updated every time, and will weight this action, k, more heavily if the reward is large, or if the probability of being picked was low and gave a moderate reward. By dividing the reward by the probability of being picked, one uses an unbiased estimator to update the weight. Unbiased estimators are explained in section 2.3.2.

Algorithm 4 EXP3

```
1: $\mathbf{w}^{(1)} \leftarrow \{w_k^{(1)} = 1\}_{k=1}^{K}$                         ▷ Weight initialization
2: for $t = 1, \cdots, T$ do
3:     $\mathbf{p}^{(t)} = (1-\gamma)\frac{\mathbf{w}^{(t)}}{\sum_k w_k^{(t)}} + \frac{\gamma}{K}$      ▷ probability over actions
4:     $k \sim$ MULTINOMIAL $(\mathbf{p}^{(t)})$                                 ▷ randomly choose action
5:     $a^{(t)} = a_k$                                                 ▷ take action
6:     RECEIVE $(r^{(t)} \in \{0,1\})$                                   ▷ Receive reward
7:     $w_k^{(t+1)} = w_k^{(t)} \exp(\frac{\gamma}{K} * \frac{r^{(t)}}{p_k^{(t)}})$                   ▷ Weight update
8: end for
```

Moving ahead, the key question is that if EXP3 is in an adversarial environment, can the learner have no regret? The adversary can know the probability of pulling each arm and the reward that arm gives, however the adversary will not know which arm the learner is choosing to pull. The same strategy that has been used in earlier classes will be used here to define an upper regret bound for the EXP3 algorithm. First, define the potential function, then define the upper and lower bounds of that potential function, combine the bounds and do some algebra to discover the regret bound.

This algorithm looks very similar to the hedge algorithm (3) that has been covered in previous lectures. One key difference is that EXP3 has partially observed rewards of the results, leading to only knowing one gain/loss for one arm and only updating one weight, as opposed to hedge's fully observed rewards. To analyse this algorithm we will convert the partially observed gain function into a fully observed gain function using an unbiased estimate, and turn the existing problem into a prediction problem with no expert advice. Starting with a partially observed gain function where only the reward from the kth arm is known, one can represent the gain function as a fully observed gain function by dividing the reward by the probability that the arm was picked. The resulting gain function is all zeros except for the kth arm where there is an unbiased estimate of the reward, $\frac{r^{(t)}}{p_k^{(t)}}$.

By doing this conversion, the regret bound can be analyzed as a expert advice problem without experts. This changes the weight update in Algorithm 4 to $w_k^{(t+1)} = w_k^{(t)} \exp(\frac{\gamma}{K} * g_k^{(t)}) \forall k$. This works because when the gain is 0, the weight will stay the same, but when there is a reward the weight will update. Lastly the probability of each action in Algorithm 4 will also change, to $\mathbf{p}^{(t)} = \frac{\mathbf{w}^{(t)}}{\sum_k w_k^{(t)}}$. These changes are vitally important and should be remembered going forward.

2.4.1 Potential Function

Proof. Lets define the potential function $\Phi^{(t)}$ for EXP3 as follows:

$$\Phi^{(t)} = \sum_{t=1}^{T} \log\big(\frac{z^{(t+1)}}{z^{(t)}}\big) \tag{36}$$

where, $z^{(t)}$ is the sum of the weights for time step t.

2.4.2 Upper Bound

Theorem 1. *(Upper bound of EXP3) This potential function can be upper and lower bounded over time, eventually the lower and upper bounds will have terms that combine to form the definition of regret, and the regret bound is built from there. For now the upper bound must be calculated.*

$$\Phi^{(t)} = \sum_{t=1}^{T} \log\big(\frac{z^{(t+1)}}{z^{(t)}}\big) \tag{37}$$

Starting from the potential function we replace z with its definition

$$\Phi^{(t)} = \sum_{t=1}^{T} \log\big(\frac{\sum_k w_k^{(t+1)}}{\sum_k w_k^{(t)}}\big) \tag{38}$$

Then insert the update equation, line 8 of algorithm 4 for $w_k^{(t+1)}$

$$\Phi^{(t)} = \sum_{t=1}^{T} \log\big(\frac{\sum_k w_k^{(t)} * \exp(\frac{\gamma}{K} * g_k^{(t)})}{\sum_k w_k^{(t)}}\big)\big) \tag{39}$$

Assume a single sequence is known.

$$\Phi^{(t)} = \sum_{t=1}^{T} \log\big(\sum_k \frac{w_k^{(t)}}{\sum_k w_k^{(t)}} * \exp(\frac{\gamma}{K} * g_k^{(t)})\big) \tag{40}$$

Replace $\frac{w_k^{(t)}}{\sum_k w_k^{(t)}}$ with the probability of selection of that arm, line 3 of algorithm 4.

$$\Phi^{(t)} = \sum_{t=1}^{T} \log\big(\sum_k \frac{p_k^{(t)} - \frac{\gamma}{K}}{1-\gamma} * \exp(\frac{\gamma}{K} * g_k^{(t)})\big) \tag{41}$$

Use a known inequality of $e^x \leq 1 + x + x^2$ to bound the function.

$$\Phi^{(t)} \leq \sum_{t=1}^{T} \log\big(\sum_k \frac{p_k^{(t)} - \frac{\gamma}{K}}{1-\gamma} * (1 + \frac{\gamma}{K} * g_k^{(t)} + (\frac{\gamma}{K} * g_k^{(t)})^2)\big) \tag{42}$$

Multiplying through

$$\Phi^{(t)} \leq \sum_{t=1}^{T} \log\big(\sum_k \frac{p_k^{(t)} - \frac{\gamma}{K}}{1-\gamma} + \sum_k \frac{p_k^{(t)} - \frac{\gamma}{K}}{1-\gamma} \frac{\gamma}{K} * g_k^{(t)} + \sum_k \frac{p_k^{(t)} - \frac{\gamma}{K}}{1-\gamma} (\frac{\gamma}{K} * g_k^{(t)})^2\big) \tag{43}$$

$$\Phi^{(t)} \leq \sum_{t=1}^{T} \log\big(1 + \frac{\gamma}{K(1-\gamma)} \sum_k p_k^{(t)} * g_k^{(t)} + \frac{\gamma^2}{K^2(1-\gamma)} \sum_k p_k^{(t)} * g_k^{(t)2}\big) \tag{44}$$

Then bounding with a known equality $\ln(1+x) \leq x$*, we get the final upper bound.*
Φ *is upper-bounded as:*

$$\Phi \leq \sum_t \big(\frac{\gamma}{K(1-\gamma)} \sum_k p_k^{(t)} * g_k^{(t)} + \frac{\gamma^2}{K^2(1-\gamma)} \sum_k p_k^{(t)} * g_k^{(t)2}\big)$$

Therefore, we have obtained the upper bound of the potential function.

2.4.3 Lower Bound

Theorem 2. *(Lower bound of EXP3) With the upper bound calculated, all that is needed is the lower bound to calculate the regret bound. Starting from the potential function, one needs to telescope the series.*

$$\Phi^{(t)} = \sum_{t=1}^{T} \log\big(\frac{z^{(t+1)}}{z^{(t)}}\big) \tag{45}$$

To telescope a series means to $\sum(x_{n+1} - x_n) = x_N - x_1$ *thus moving the series of subtractions to the outer most extremes. In this case it would look like this.*

$$\Phi^{(t)} = \log\big(\frac{\sum_k w_k^{(T)}}{\sum_k w_k^{(1)}}\big) \tag{46}$$

Then from line 2 of Algorithm 4, we know that each weight at time step 1 is 1, so:

$$\Phi^{(t)} = \log\big(\frac{\sum_k w_k^{(T)}}{K}\big) \tag{47}$$

Separate the logs and apply the update equation from line 8 of algorithm 4.

$$\Phi^{(t)} = -\log(K) + \log(\sum_k w_k^{(T)} \exp(\frac{\gamma}{K} * g_k^{(t)}) \tag{48}$$

Recursively apply the update equation from line 8 of algorithm 4.

$$\Phi^{(t)} = -\log(K) + \log(\sum_{k=1}^{K} \prod_{t=1}^{T} \exp(\frac{\gamma}{K} * g_k^{(t)}) \tag{49}$$

Then one can switch the product term into a sum by pushing it into the exponent

$$\Phi^{(t)} = -\log(K) + \log(\sum_{k=1}^{K} \exp(\frac{\gamma}{K} \sum_{t=1}^{T} g_k^{(t)}) \tag{50}$$

Any element lower bounds a sum

$$\Phi^{(t)} \geq -\log(K) + \log(\exp(\frac{\gamma}{K} \sum_{t=1}^{T} g_k^{(t)}) \forall k \tag{51}$$

Algebra gets the equation into the final lower bound.

$$\Phi^{(t)} \geq -\log(K) + \frac{\gamma}{K} \sum_{t=1}^{T} g_k^{(t)} \forall k$$

2.4.4 Combine Bounds

Finally, the lower and upper bounds can be combined to discover the regret bound.

$$-\log(K) + \frac{\gamma}{K} \sum_{t=1}^{T} g_k^{(t)} \leq \sum_t \big(\frac{\gamma}{K(1-\gamma)} \sum_k p_k^{(t)} * g_k^{(t)} + \frac{\gamma^2}{K^2(1-\gamma)} \sum_k p_k^{(t)} * g_k^{(t)2}\big) \tag{52}$$

Then by rearranging terms and dividing each side by $\frac{\gamma}{K}$

$$\sum_{t=1}^{T} g_k^{(t)} - \frac{1}{1-\gamma} \sum_t \sum_k p_k^{(t)} * g_k^{(t)} \leq \frac{K}{\gamma} \log(K) + \frac{\gamma}{K(1-\gamma)} \sum_t \sum_k p_k^{(t)} * g_k^{(t)2} \tag{53}$$

Rearrange more terms

$$(1-\gamma) \sum_{t=1}^{T} g_k^{(t)} - \sum_t \sum_k p_k^{(t)} * g_k^{(t)} \leq \frac{K}{\gamma} \log(K) + \frac{\gamma}{K} \sum_t \sum_k p_k^{(t)} * g_k^{(t)2} \tag{54}$$

$$\sum_{t=1}^{T} g_k^{(t)} - \sum_t \sum_k p_k^{(t)} * g_k^{(t)} \leq \frac{K}{\gamma} \log(K) + \frac{\gamma}{K} \sum_t \sum_k g_k^{(t)} + \gamma \sum_{t=1}^{T} g_j^{(t)} \tag{55}$$

Take the expectation of both sides

$$\mathbb{E}_p[\sum_{t=1}^{T} g_k^{(t)}] - \mathbb{E}_p[\sum_t \sum_k p_k^{(t)} * g_k^{(t)}] \leq \frac{K}{\gamma} \log(K) + \frac{\gamma}{K} \sum_t \sum_k \mathbb{E}_p[g_k^{(t)}] + \gamma \sum_{t=1}^{T} \mathbb{E}_p[g_j^{(t)}] \tag{56}$$

Simplify expectations into other known terms

$$\sum_{t=1}^{T} r_j^{(t)} - \mathbb{E}_p[\sum_t \sum_k p_k^{(t)} * g_k^{(t)}] \leq \frac{K}{\gamma} \log(K) + \frac{\gamma}{K} \sum_t \sum_k \mathbb{E}_p[r_k^{(t)}] + \gamma \sum_{t=1}^{T} \mathbb{E}_p[r_j^{(t)}] \tag{57}$$

Replace rewards with gains

$$G_{max} - \mathbb{E}_p[\sum_t \sum_k p_k^{(t)} * g_k^{(t)}] \leq \frac{K}{\gamma}\log(K) + \frac{\gamma}{K}K(G_{max}) + \gamma G_{max} \tag{58}$$

Simplify

$$G_{max} - \mathbb{E}_p[\sum_t \sum_k p_k^{(t)} * g_k^{(t)}] \leq \frac{K}{\gamma}\log(K) + 2\gamma G_{max} \tag{59}$$

The left term is equivalent to the expected regret of the function, and the right side can be further reduced by finding an optimal value for γ. This can be done by taking the derivative of the right side and solving for γ.

$$\frac{d}{d\gamma}(\frac{K}{\gamma} + 2\gamma T) = 0 \tag{60}$$

$$-\frac{K\log K}{\gamma^2} + 2T = 0 \tag{61}$$

$$\frac{K\log K}{\gamma^2} = 2T \tag{62}$$

$$\frac{K\log K}{2T} = \gamma^2 \tag{63}$$

$$\sqrt{\frac{K\log K}{2T}} = \gamma \tag{64}$$

Then by plugging in this γ one can simplify the result to get a nice expected regret bound.

$$\mathbb{E}[R^{(T)} \leq \frac{K\log K}{\gamma} + 2\gamma T \tag{65}$$

$$\mathbb{E}[R^{(T)} \leq \frac{K\log K}{\sqrt{\frac{K\log K}{2T}}} + 2\sqrt{\frac{K\log K}{2T}}T \tag{66}$$

$$\mathbb{E}[R^{(T)} \leq \sqrt{2TK\log K} + \sqrt{2TK\log K} \tag{67}$$

$$\mathbb{E}[R^{(T)} \leq 2\sqrt{2TK\log K} \tag{68}$$

$$\mathbb{E}[R^{(T)} \leq O(\sqrt{TK\log K}) \tag{69}$$

$$\tag{70}$$

Thus the regret is bounded by T and K. This means that EXP3 is a no regret algorithm! This regret algorithm is dependent on a non-stochastic environment, or even an adversarial environment, and is compared to the best performing arm. If the environment was stochastic, then the upper bound would be smaller, and the solve would be different due to making different assumptions.

2.5 EXP4

The EXP3 algorithm is named as such because of the three behaviors that define this algorithm, exploitation, exploration, and exponential-weight updating. EXP4, the next algorithm learned in *this class, adds one more exp, expert advice. The algorithm is similar in most ways and uses the context of experts to make better decisions. In this way, the algorithm has a slightly smaller upper regret bound.*

Algorithm 5 EXP4

```
1: $\mathbf{w}^{(1)} \leftarrow \{w_n^{(1)} = 1\}_{n=1}^{N}$                          ▷ Weight initialization
2: for $t = 1, \cdots, T$ do
3:     Receive$\mathbf{X}^{(t)} \in \mathbb{R}^N$                                    ▷ advice from experts
4:     $\mathbf{p}^{(t)} = (1-\gamma)\frac{\mathbf{w}^{(t)}}{w_k^{(t)}} * \mathbf{X}^{(t)} \in \Delta^K$            ▷ probability over actions
5:     $k \sim$Multinomial $(\mathbf{p}^{(t)})$                                     ▷ randomly choose action
6:     Receive $(r^{(t)})$                                                  ▷ Receive reward
7:     $^{(t)} = \frac{r^{(t)}}{q_k^{(t)}\mathbb{I}[k=k^{(t)}]\in\mathbb{I}^K}$                          ▷ reward over all terms
8:     $\mathbf{g}^{(t)} = \mathbf{X}^{(t)} * \hat{\mathbf{r}}^{(t)} \in \mathbb{R}^N$                             ▷ Per expert reward
9:     $w_k^{(t+1)} = w_k^{(t)} \exp(\gamma * g_n^{(t)}) \forall n$                          ▷ Weight update
10: end for
```

There are some key differences between EXP4 and EXP3. EXP4 relies on expert advice to inform the weights, as shown on line 5 of EXP4 in comparison to EXP3's line 4. Then the expert advice also informs the rewards that update the weights, lines 10 and 7 for comparison. These differences do not vary the derivation of the regret bound enough to warrant re-deriving it. What ends up changing is that the regret of EXP4 is as follows.

$$R_{EXP4} \leq \sqrt{KT \log N} \tag{71}$$

This is true where K is the number of arms, T is the total number of update steps, and N is the number of experts. This algorithm can make a difference over EXP3 when there is a very large number of arms, and a limited number of experts that can inform the learner on each arm, otherwise EXP3 and EXP4 perform similarly.

References

[1] Kitani, Kris. "Bandit: EXP3." Statistical Techniques in Robotics. Statistical Techniques in Robotics, 24 Mar. 2021, Pittsburgh, PA. www.dropbox.com/sh/ot8ijbxfr9ngol2/AAAruOFyQu3T6582fWm1Aumja?dl=0preview=L14 EXP3-EXP4-Review.pdf.

[2] Kitani, Kris. "Thompson Sampling" Statistical Techniques in Robotics. Statistical Techniques in Robotics, 17 Mar. 2021, Pittsburgh, PA. https://www.dropbox.com/sh/ot8ijbxfr9ngol2/AAAruOFyQu3T6582fWm1Aumja?dl=0preview=L13+MAB-Thompsons+Sampling.pdf